CS230

Consistent Video Colorization

Gael Colas
Stanford University
Department of Aeronautics
colasg@stanford.edu

Kevin Lee
SCPD
Department of
Electrical Engineering
kelelee@stanford.edu

Rafael Rafailov
Stanford University
Department of Statistics
rafailov@stanford.edu

Abstract

In this paper we try to solve the problem of consistent video colorization. We explore the use of a hybrid Supervised/Conditional Generative Adversarial Networks (cGAN) to color gray-scale videos. The generator is conditioned on the current grayscale image and previous colorized frames to propagate color forward to the new frame. Since the goal of video colorization is consistency and plausibility rather than exact accuracy, we believe that we can achieve superior results using GANs, as compared to a stand-alone supervised model.

1 Introduction

The goal of this project is to build a system for grayscale video colorization. The problem of image colorization has been studied extensively in deep learning and machine learning in general and currently several models exist that achieve very good results on this task. One might guess that a natural approach for video colorization would be to color each frame separately and then concatenate them back into a video. There is however a major drawback to this approach: we are not guaranteed to (and in fact do not) get consistent colors between frames. Our goal is to develop a model that can propagate a consistent coloring scheme between different frames of a video.

One input of the colorizing pipeline X is a **grayscale video** sampled at 30 frames per second (fps). The number of frames of an input T depends on the duration of the video. One label Y is the corresponding original color video sampled at $30fps$. We process input frames in the shape: $(256, 256, 1)$ and the corresponding output has the shape: $(256, 256, 3)$. Currently we only try to colorize the video a single frame at a time, propagating color from a colourized frame from $t-1$ to a grayscale frame at t.

2 Related work

There are some methods developed for "Automatic Video Colorization" outside of the Deep Learning field. For example [ZYL12] used supervised and unsupervised learning techniques as well as optimization techniques to colorize a video clip. Their algorithm takes as input a grayscale image and uses an image with a similar content to colorize the input image. The colors of a region in the reference image are used as a *scribble* to color the corresponding region in the grayscale image. [LLW04] introduced this idea of "Image Colorization" from *scribbles* of an expert. The expert draws some lines of a given color over different regions of the image. Then these colors are propagated over the whole image, based on the idea that neighbouring pixels with similar intensities should have similar colors. This is achieved by solving a quadratic minimization problem. All of these methods require significant supervision and involvement, and are in general quite tedious in order to produce high-quality results.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Figure 1: Example of an input-label pair from the "baking" dataset (images are $dt = 5$ frames apart) Top: input = grayscale video; Bottom: label = corresponding original color video

Figure 2: Baseline Video Colorization on two examples: "ocean" and "horses" (images are $dt = 5$ frames apart) (a) color ground truth, (b) baseline frame-by-frame colorization

In the Deep Learning field, [ZIE16] designed a Convolutional Neural Network (CNN) to learn how to output a *plausible* colorization from a grayscale input image, without any reference image. Very recently [HCL$^+$18] have proposed the first deep learning approach for exemplar-based local colorization. Their model uses a reference image (with features that can be different from the grayscale image) to colorize a grayscale image. The model involves two stages with separate neural nets. First it computes the similarities between the two images with a "similarity sub-net". Then it propagates the color from the reference image to the grayscale image using the computed similarity map, with a "colorization sub-net". This model can indeed be used for propagating color between separate frames and get consistent video colorization. We have done some work on implementing this as a baseline, but it seems that the build is somewhat unstable and we were not able to get meaningful results out of the model. In a preliminary paper [VSF$^+$18] the authors deploy a somewhat similar idea. They built a convolutional to encoding images and assign colors to video frames based on encoding similarity.

Our project builds mostly on the work made by [NNE18] that uses conditional GANs for image colorization. In particular we extend the model to achieve color propagation.

3 Dataset and Features

We leveraged the "Moments In Time" dataset by MIT-IBM Watson AI Lab [Lab]. This dataset includes a collection of one million labeled 3 second videos, involving people, animals, objects or natural phenomena, that capture the gist of a dynamic scene. It is divided into subcategories depending on the nature of the activity displayed. We specifically used the "baking" subcategory to train our model.

The videos were randomly split between train, validation and test set, composed respectively of 450, 50 and 50 videos (80% / 20% / 20% split) and then converted to a consistent height ($H = 256$), width ($W = 256$) and fps ($fps = 30$) with the Open CV Python package [pac17]. Figure 1 shows an example of an input-label pair.

From each video, several examples were extracted as follows: one example is the concatenation of the grayscale image from time t and the color image from time $(t - dt)$. A label is the corresponding color image from time t. After pre-processing, our training, validation and test set respectively consisted of about **7 000**, 2 800 and 2 800 samples.

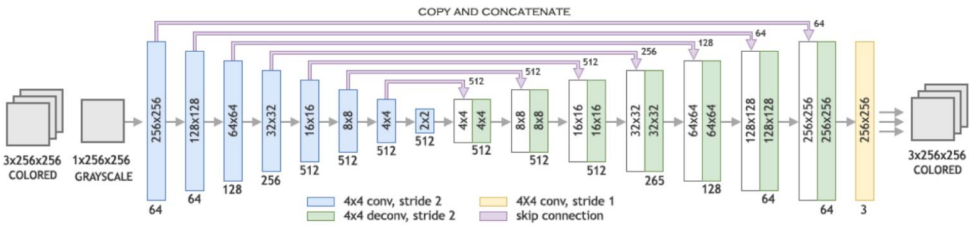

Figure 3: Baseline GAN architecture for the generator model

4 Methods

4.1 Baseline

Our baseline method is to apply *naively* the model from [ZIE16] to every frame of the video independently. To use it in our problem, we had to build a front-end addition to their model to handle videos instead of images. Some results are presented in Figure 2. As can be seen coloring is quite consistent when there is little change in the landscape (the ocean example) even though not quite perfect. However, we see significant changes in more dynamic environments, i.e. the hide of the running horse moving from blue to red. This is the main issue we try to tackle with our model.

4.2 Hybrid Conditional GAN

Our main method builds on the work from [NNE18]. In this paper, the authors used a conditional GAN to colorize grayscale images. The conditioning set was a single grayscale image and the goal was to generate a plausible colorization.

The core goal of our approach was to expand the above model to color propagation, rather than just color generation. We achieve that by expanding the conditioning set to include the previous colored frame. More precisely, an input (condition set) to the generator network is a grayscale frame from time t: g_t and the corresponding color frame from time $(t-1)$: c_{t-1}. The model then tries to create a colorized version $\hat{c}_t$ from g_t using the colors of c_{t-1}. Mathematically, the objective function of the generator is:

$$\min_{\theta_G} -\alpha \mathbb{E}_{\mathbf{z}}[\log D(G(\mathbf{0}_z|[c_{t-1}, g_t])|c_{t-1})] + \beta||G(\mathbf{0}_z|[c_{t-1}, g_t]) - c_t||_1$$

Notice that this is a hybrid supervised/C-GAN model. For $\alpha = 0$ this reduces to a classical supervised coloring algorithm, while for $\beta = 0$ gives us a C-GAN. We train two models: a direct supervised model ($\alpha = 0$) and one with $\alpha = 1$, $\beta = 100$. The main idea here is that since we prioritize plausible color propagation, versus exact accuracy, the C-GAN block can act as a regularizer to the supervised algorithm. Here we use the conditional GAN non-saturating loss. The input of the network is treated as zero noise $z = 0$ with c_{t-1} and g_t as prior. Unlike the traditional GAN, this model does not generate from random noise as we would not like to introduce variability in the obtained colorization.

Our generator model architecture is based around **U-net** and is presented in Figure 3 (figure adapted from [NNE18]). The architecture is symmetric with 7 encoding and 7 decoding layers. Unlike the original U-net model, however 1.) all max pooling layers have been replaced by strided convolutions 2.) ReLU activations are replaced with with Leaky ReLUs and 3). every convolutional layer is followed by a batch normalization layer, as recommended by [AR15].

The discriminator does the following mapping: $D : (c_t, c_{t-1}) \rightarrow p(c_t, c_{t-1})$. It tries to distinguish between real and fake colorized next frames based on the previous color frame. It has a standard architecture for a classification CNN - several convolutional layers followed by batch normalization layers. Mathematically, the objective is:

$$\max_{\theta_D} \mathbb{E}_{c_t}[\log D(c_t|c_{t-1})] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{0}_z|[c_{t-1}, g_t])|c_{t-1}))]$$

Model		sup Gen	cGAN ($dt = 1$)	cGAN ($dt = 5$)
Pixel Accuracy (%)	Train	94	92	84
	Test	78	83	79

Table 1: Pixel Accuracy comparison between the supervised model and different Conditional GAN models

Two different version of this model were trained: the first one was trained on a dataset composed of directly consecutive frames ($dt = 1$), the other one was trained on more time-spaced consecutive frames ($dt = 5$). Finally, a front-end program was built to handle grayscale video colorization. It takes as input a grayscale video and the first re-colorized frame. To test the color propagation, we used the ground truth as the first re-colorized frame. However, to build an end-to-end video colorization pipeline, the first frame can be re-colorized using the image colorization network from [ZIE16].

5 Experiments/Results/Discussion

To **quantitatively** compare our models, we use the pixel accuracy metric. This is the proportion of pixel values in the re-colorized image which have the same value as the ground truth.

Table 1 compares the pixel accuracy of the different models on both the train and test set. It is important to precise that this was evaluated on the "color transfer" task: how well the algorithm colorizes a grayscale image given the previous ground truth color frame. First we see, that the supervised model achieves the highest pixel accuracy on the train set ($dt = 5$) on par with the GAN model for $dt = 1$. As expected, the conditional GAN trained on directly consecutive frames ($dt = 1$) performs better according to this metric than the one trained on more time-spaced frames ($dt = 5$). This makes sense because it is easier to predict the color of directly consecutive frames: just copy-pasting the color gives a sensible result. Thus this metric is not a good estimator of the quality of the samples. One interesting thing to note here is that, when we trained the supervised model (we performed gradient descent only on the L1 norm loss), the discriminator loss converges to zero, while the generator loss keeps going higher, which is not the case when training the hybrid model. Essentially, training only based on the L1 norm creates some sort of coloring anomalies, that the discriminator easily picks up and can always identify colorized images from original ones. This is not the case when training with the additional GAN objective. Indeed, after 10 epochs of training the discriminator is unable to distinguish colorized frames from original video ones, which we interpret as the GAN block having a smoothing effect on the coloring procedure.

We also considered **qualitative metrics** when evaluating the results of our model. The flaws in the outputs of different models were easy to spot when propagating through entire videos, so we could visually rank our algorithms. They were run on a the validation "baking" video set and compared on the same videos. Figure 4 presents the colorization results of our different algorithms on the same validation video. The images in the figure were sampled from the video every 7 frames.

The second video (second row) show the result of using th supervised model to perform color propagation. This model was trained only on the L_1-distance bewteen the re-colorized and the original color frame. This metric does not encode any judgment over the quality of the samples: no discriminator to judge over the plausibility of the colorization. In this framework, copy-pasting colors gives a sensible result. This is what the network does: we can see for example that the grand-mother's appron colors do not move with it. This causes the colors of the video to quickly go bad.

To generate the third video (third row), our conditional GAN model was applied to each grayscale frame conditioned on the previous ground truth color frame. This re-colorized video is almost indistinguishable from the ground truth. This shows that the models achieve good color transfer between frames. However, as seen on the next rows, the performances of the GAN models G quickly decline when using them for color propagation over the whole video. The videos were colorized step-by-step as follows: at time t, the grayscale frame g_t was colorized with $\hat{c}_t = G(g_t, \hat{c}_{t-1})$ (or $\hat{c}_t = G(g_t, \hat{c}_{t-1}, c_1)$ for the extended conditional GAN model), where $\hat{c}_{t-1}$ is the previous re-colorized frame.

The models used to generate the fourth, fifth are our conditional GAN model trained with ($dt = 1$) consecutive frames (d) and our conditional GAN model trained with ($dt = 5$) consecutive frames

Figure 4: Comparison of Video Colorizations. From top row to bottom: (a) baseline frame-by-frame colorization, (b) supervised model color propagation, (c) GAN colorization from ground truth reference, (d) GAN color propagation (trained with $dt = 1$), (e) GAN color propagation (trained with $dt = 5$)

(f=e). The first issue is that consecutive frames are very highly correlated. When training on consecutive frames ($dt = 1$), the conditional GAN model can essentially learn to copy-paste colors to achieve a very good pixel accuracy. However, when we apply color propagation throughout a video, this causes colorization errors on fast moving objects. This can be seen on the second image of the fourth row. Even tough the left hand of the front girl moved, the algorithm colorizes it in gray: the color of the background which occupied these pixels in the first frame.

To try to solve this issue, our network was retrained on more time-spaced frames: ($dt = 5$) consecutive frames. This helped as can be seen on the second image of the fifth row. This time, the algorithm managed to track the hand of the girl which is now correctly colorized in pink.

However, this approach still could not solve the problem of **exponential error propagation**. This error is caused by the distribution difference between training examples and what the GAN has to work with for color propagation. Indeed, at test time, the conditional GAN must colorize the grayscale frame g_t conditioned on the ground truth color frame c_{t-dt}. However when doing color propagation, the GAN must colorize the grayscale frame g_t conditioned on the previous re-colorized frame $\hat{c}_{t-1}$. This means that if there are some mistakes in the colorization of frame $\hat{c}_{t-1}$, this wrong colorization will be considered as the ground truth to colorize the next frame. This can be seen for example in the fifth row where the blue color mistake gets amplified and contaminates the colorization of all neighbouring pixels.

Quantitatively if the pixel accuracy for color transfer is $(1-\eta)$, then after t steps of color propagation the pixel accuracy of the t-th frame is on the order of $(1-\eta)^t$: hence the exponential error propagation.

6 Conclusion/Future Work

We successfully trained a GAN-based model for color transfer in grayscale videos. Our model achieves high pixel accuracy, and plausibility for color transfer between consecutive frames. However this performance declines quickly when we use this model to perform end-to-end color propagation due to **exponential error propagation** throughout the video. We tried extending our model by including an extra conditioning on a color reference image, but that did not improve the performances.

The problem of video colorization has a very obvious recurrent structure. If we had more time, an idea we would have explore was to take advantage of this recurrent structure to colorize a grayscale video in a single pass. This network architecture would take advantage of the idea of convolutional LSTM layers introduced in [Yeu17] to process a sequence of images, rather than propagate color through a single frame at each pass. This approach could solve the exponential error propagation.

7 Contributions

We worked together on reviewing literature and existing GitHub repositories, studying state-of-the-art CNN and GAN models and contributing to the poster and reports. Besides the common tasks, Gael performed initial scouting of available image colorization models to narrow down the scope and performed baseline training. He built the backend pipeline to perform video colorization from the image colorization models. Kevin set up the AWS environment to work with the different models tested. He searched for available training data and did the initial data pre-processing using OpenCV [pac17]. Rafael modified the conditional GAN architecture to make it suitable for our problem. He performed model training, and parameter tuning on these models. We all performed performances analysis of our models, these discussions led to the model modifications outlined in the report.

8 Codebase

All our work is available on the following GitHub repository: `https://github.com/ColasGael/Automatic-Video-Colorization`

References

[AR15] Soumith Chintala Alec Radford, Luke Metz. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015. https://arxiv.org/abs/1511.06434.

[HCL+18] Mingming He, Dongdong Chen, Jing Liao, Pedro V. Sander, and Lu Yuan. Deep exemplar-based colorization. *ACM Trans. Graph.*, 37(4):47:1–47:16, July 2018.

[Lab] MIT-IBM Watson AI Lab. "moments in time" dataset. http://moments.csail.mit.edu/.

[LLW04] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, August 2004.

[NNE18] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization using generative adversarial networks. In *International Conference on Articulated Motion and Deformable Objects*, pages 85–94. Springer, 2018.

[pac17] Open CV Python package. Open cv - getting started with videos, 2017. https://docs.opencv.org/3.4.0/dd/d43/tutorial_py_video_display.html.

[VSF+18] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. 2018. https://arxiv.org/pdf/1806.09594.

[Yeu17] Xingjian Shi Zhourong Chen Hao Wang Dit-Yan Yeung. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *NIPS 2017*, 2017.

[ZIE16] Richard Zhang, Phillip Isola, and Alexei Efros. Colorful image colorization. 9907:649–666, 10 2016. https://github.com/richzhang/colorization.

[ZYL12] Zhong Zhen, Gui Jun Yan, and Ma Lizhuang. An automatic image and video colorization algorithm based on pattern continuity. *2012 International Conference on Audio, Language and Image Processing*, pages 531–536, 2012.

9 CS 236 Project

This was a combined project with CS 236. We specifically trained the supervised color propagation algorithm for CS 230 and the conditional GAN model was shared between both projects. In addition fro CS 236 we trained models in additional set-ups by including base-reference images in the conditioning set and experimenting with drop-out layers.

Video Colorization using GANs

Gael Colas
Stanford University
Department of Aeronautics and Astronautics
colasg@stanford.edu

Rafael Rafailov
Stanford University
Department of Statistics
rafailov@stanford.edu

Abstract

In this paper we explore the use of Conditional Generative Adversarial Networks (cGAN) to color gray-scale videos. We modify a U-net based architecture and condition on the current grayscale image and previous colorized frames to propagate color forward to the new frame. Since the goal of video colorization is consistency and plausibility rather than exact accuracy, we believe that we can achieve superior results using GANs, as compared to a stand-alone supervised model.

1 Introduction

The goal of this project is to build a system for grayscale video colorization. The problem of image colorization has been studied extensively in deep learning and machine learning in general and currently several models exist that achieve very good results on this task. One might guess that a natural approach for video colorization would be to color each frame separately and then concatenate them back into a video. There is however a major drawback to this approach: we are not guaranteed to (and in fact do not) get consistent colors between frames. Our goal is to develop a model that can propagate a consistent coloring scheme between different frames of a video.

2 Related Work

There are some methods developed for "Automatic Video Colorization" outside of the Deep Learning field. For example [ZYL12] used supervised and unsupervised learning techniques as well as optimization techniques to colorize a video clip. Their algorithm takes as input a grayscale image and uses an image with a similar content to colorize the input image. The process is as follows: the reference image is segmented into regions and each region is given a class number. Every pixel of the reference image is mapped into a high dimensional 'Gabor wavelet' feature vector. Then, PCA is applied to maximize the distance between the classes. kNN is applied to the grayscale image pixels to assign them a class number. The regions are smoothed using Spatial Consistency. Finally, the colors of a region in the reference image are used as a *scribble* to color the corresponding region in the grayscale image. [LLW04] introduced this idea of "Image Colorization" from *scribbles* of an expert. The expert draws some lines of a given color over different regions of the image. Then these colors are propagated over the whole image, based on the idea that neighbouring pixels with similar intensities should have similar colors. This is achieved by solving a quadratic minimization problem. All of these methods require significant supervision and involvement, and are in general quite tedious in order to produce high-quality results.

In the Deep Learning field, [ZIE16] designed a Convolutional Neural Network (CNN) to learn how to output a *plausible* colorization from a grayscale input image, without any reference image. Very recently [HCL+18] have proposed the first deep learning approach for exemplar-based local colorization. Their model uses a reference image (with features that can be different from the

Preprint. Work in progress.

Figure 1: Example of an input-label pair from the "baking" dataset (images are $dt = 5$ frames apart) Top: input = grayscale video; Bottom: label = corresponding original color video

grayscale image) to colorize a grayscale image. The model involves two stages with separate neural nets. First it computes the similarities between the two images with a "similarity sub-net". Then it propagates the color from the reference image to the grayscale image using the computed similarity map, with a "colorization sub-net". This model can indeed be used for propagating color between separate frames and get consistent video colorization. We have done some work on implementing this as a baseline, but it seems that the build is somewhat unstable and we were not able to get meaningful results out of the model. In a preliminary paper [VSF+18] the authors deploy a somewhat similar idea. They built a convolutional to encoding images and assign colors to video frames based on encoding similarity.

Our project builds on the work made by [NNE18] that uses conditional GANs for image colorization. In particular we extend the model to achieve color propagation.

3 Task Definition

Given a grayscale video $X \in \mathbb{R}^{T \times H \times W \times 1}$, our objective is to learn the mapping $\hat{Y} = \mathcal{F}(X)$ to the associated color video $Y \in \mathbb{R}^{T \times H \times W \times 3}$. A grayscale video X is represented as a succession of grayscale frames: $X = \{c_t \in \mathbb{R}^{H \times W \times 1}, 1 \leq t \leq T\}$.

One input X is a grayscale video sampled at 30 frames per second (fps). The number of frames of an input T depends on the duration of the video. One reference output Y (label) is the corresponding original color video sampled at $30fps$. We process input frames in the shape: $(256, 256, 1)$ and the corresponding output has the shape: $(256, 256, 3)$. However, as our model is fully convolutional, it can handle images of varying sizes. Figure 1 shows an example of an input-output pair.

4 Dataset

We leveraged a readily available public dataset to source videos for our project: the "Moments In Time" dataset by MIT-IBM Watson AI Lab [Lab]. This dataset includes a collection of one million labeled 3 second videos, involving people, animals, objects or natural phenomena, that capture the gist of a dynamic scene. It is divided into subcategories depending on the nature of the activity displayed. We specifically used the "baking" subcategory to train our model.

The videos were randomly split between train, validation and test set, composed respectively of 450, 50 and 50 videos (80% / 20% / 20% split) and then converted to a consistent height ($H = 256$), width ($W = 256$) and fps ($fps = 30$) with the Open CV Python package [pac17].

5 Approaches

5.1 Baseline

Our baseline method is to apply *naively* the model from [ZIE16] to every frame of the video independently. This Neural Network was already trained on the ImageNet dataset and the model weights are available on the associated GitHub repository. To use it in our problem, we had to build a

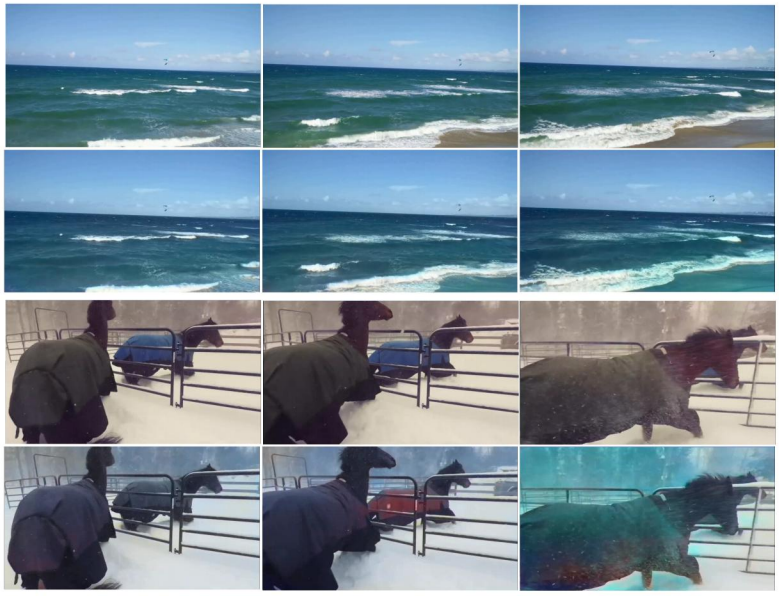

Figure 2: Baseline Video Colorization on two examples: "ocean" and "horses" (images are $dt = 5$ frames apart)
From top row to bottom: (a) "ocean" color ground truth, (b) "ocean" baseline frame-by-frame colorization, (c) "horses" color ground truth, (d) "horses" baseline frame-by-frame colorization

front-end addition to their model to handle videos instead of images. Some results are presented in Figure 2:

As can be seen coloring is quite consistent when there is little change in the landscape (the ocean example) even though not quite perfect. However, we see significant changes in more dynamic environments, i.e. the hide of the running horse moving from blue to red. This is the main issue we try to tackle with our model.

5.2 Conditional GAN

Our main method builds on the work from [NNE18]. In this paper, the authors used a conditional GAN to colorize grayscale images. The conditioning set was a single grayscale image and the goal was to generate a plausible colorization.

The core goal of our approach was to expand the above model to color propagation, rather than just color generation. We achieve that by expanding the conditioning set to include the previous colored frame. More precisely, an input (condition set) to the GAN generator is a grayscale frame from time t: g_t and the corresponding color frame from time $(t-1)$: c_{t-1}. Then the generator tries to create a colorized version $\hat{c}_t$ from g_t using the colors of c_{t-1}. Mathematically, the objectives function of the generator is:

$$\min_{\theta_G} -\mathbb{E}_{\mathbf{z}}[\log D(G(\mathbf{0}_z|[c_{t-1}, g_t])|c_{t-1})] + \lambda||G(\mathbf{0}_z|[c_{t-1}, g_t]) - c_t||_1$$

Here we use the conditional GAN non-saturating loss. The input of the network is treated as zero noise $z = 0$ with c_{t-1} and g_t as prior. Unlike the traditional GAN, this model does not generate from random noise as we would not like to introduce variability in the obtained colorization. The L_1 penalty was added to preserve the grayscale frame content after the colorization operation.

Our generator model architecture is based around **U-net** and is presented in Figure 3 (figure adapted from [NNE18]). The architecture is symmetric with 7 encoding and 7 decoding layers. Unlike the original U-net model, however all max pooling layers have been replaced by strided convolutions and ReLU activations with Leaky ReLUs, as recommended by [AR15]. In more details:

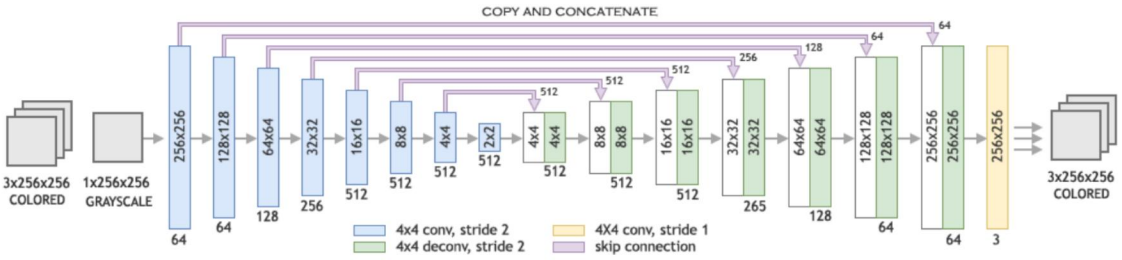

Figure 3: Baseline GAN architecture for the generator model

Figure 4: Baseline GAN architecture for the discriminator model

- The convolutional layers use filters of size 4×4 and a stride of 2. Each layer is followed by batch normalization and a leaky ReLU activation.
- Each layer in the deconvolutional part consist of 4×4 transposed convolutions and stride of 2. The layer output is then concatenated with the output of the corresponding mirror convolutional layer and batch normalization and leaky ReLU activation are applied.
- The final layer consists of 3 1×1 convolutions, i.e. one for each color channel of the generated image.

The discriminator does the following mapping: $D : (c_t, c_{t-1}) \rightarrow p(c_t, c_{t-1})$. It tries to distinguish between real and fake colorized next frames based on the previous color frame. It has a standard architecture for a classification CNN - several convolutional layers followed by batch normalization layers. Mathematically, the objective is:

$$\max_{\theta_D} \mathbb{E}_{c_t}[\log D(c_t|c_{t-1})] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{0}_z|[c_{t-1}, g_t])|c_{t-1}))]$$

We adapted the model from [NNE18] to deal with this additional conditioning. The modified network's input is a tensor obtained from concatenating c_{t-1} and c_t (or c_{t-1} and $\hat{c}_t$ if the images come from the generator); the input shape is: $6 \times 256 \times 256$. The discriminator architecture is presented in Figure 4.

Two different version of this model were trained: the first one was trained on a dataset composed of directly consecutive frames ($dt = 1$), the other one was trained on more time-spaced consecutive frames ($dt = 5$).

Finally, a front-end program was built to handle grayscale video colorization. It takes as input a grayscale video and the first re-colorized frame. To test the color propagation, we used the ground truth as the first re-colorized frame. However, to build an end-to-end video colorization pipeline, the first frame can be re-colorized using the image colorization network from [ZIE16].

5.3 Conditional GAN extension

The previous model achieves good results in terms of color transfer between frames, measured both qualitatively and qualitatively (see 6 for more details). However, when using it to colorize a whole video, the performances collapse. Indeed in this color propagation end-to-end pipeline: a re-colorized frame is then used as a reference to colorize the next frame. Small colorization errors in the first colorization lead to huge color changes after a few propagation step: this is what we call "exponential error propagation".

One way around this problem is to have a single reference colored image (instead of using the previous colored frame) and color every frame from that baseline. [HCL$^+$18] uses this approach. However, in the case of video colorization, this approach does not benefit from the high level of correlation between consecutive frames.

To try to solve this issue, this "extended" model introduce an additional conditioning on the first colorized image. The idea is to use the color of the first frame as a "good color palette" reference to dampen the exponential decay of the color error propagation.

5.4 Extra Ideas

The problem of video colorization has a very obvious recurrent structure. An idea that we did not have the time to implement was to take advantage of this recurrent structure to colorize a grayscale video in a single pass. This network architecture would take advantage of the idea of convolutional LSTM layers introduced in [Yeu17] to process a sequence of images, rather than propagate color through a single frame at each pass.

This approach could solve the exponential error propagation.

6 Results

6.1 Training

The conditional GAN architecture from [NNE18] was originally trained on CIFAR10 and Places365. However, both the Generator and the Discriminator architectures were modified and thus had to be retrained from scratch on the "baking" dataset described in 4.

From each video, several examples were extracted as follows: one example is the concatenation of the grayscale image from time t and the color image from time $(t - dt)$, and also the first color image from time $(t = 1)$ for the extended GAN model. A label is the corresponding color image from time t.

After processing, our training set consisted of about **7,000 samples**. We trained each GAN model for **10 epochs**.

6.2 Performance metrics

We used two different **quantitative metrics**:

- L_1 difference between the ground truth and the re-colorized images. This is the same as the penalty presented in 5.2 to ensure content preservation;
- pixel accuracy: this is the proportion of pixel values in the re-colorized image which have the same value as the ground truth (after post-processing and discretization).

There is an inherent issue with both metrics: we are interested in getting **plausible** consistent colorization, rather than matching exactly the original one. For example, we do not care whether the color of the horse's hide is red or blue, as long as it remains consistent throughout the video.

That is why we considered **qualitative metrics** when evaluating the results of our model. The flaws in the outputs of different models were easy to spot, so we could visually ranked our algorithms. They were run on a the validation "baking" video set and compared on the same videos.

If our results had been better and it had been more difficult to visually rank our algorithms' performances, we thought about using what is called the "Colorization Turing Test". In this evaluation

Model		cGAN ($dt = 1$)	cGAN ($dt = 5$)	Extended cGAN ($dt = 5$)
Pixel Accuracy (%)	Train	92	84	67
	Validation	83	79	63

Table 1: Pixel Accuracy comparison between the different Conditional GAN models

framework, independent people are asked to watch videos. They must tell which one has been generated: i.e. which one has been re-colorized using a neural network. If the evaluation panel fails around 50% of the time, this means that people are randomly guessing: they cannot distinguish between real and fake color video. This qualitative evaluation was first suggested by [ZIE16] to evaluate plausible image colorization. [NNE18] provides an application for running the tests.

6.3 Evaluation

Table 1 compares the pixel accuracies of the different Conditional GAN models on both the train and validation set. It is important to precise that this was evaluated on the "color transfer" task: how well the algorithm colorizes a grayscale image given the previous ground truth color frame. We can see that the conditional GAN trained on directly consecutive frames ($dt = 1$) performs better according to this metric than the one trained on more time-spaced frames ($dt = 5$). This makes sense because it is easier to predict the color of directly consecutive frames: just copy-pasting the color gives a sensible result. Thus this metric is not a good estimator of the quality of the samples.

The evaluation of how well the algorithm performs on "color propagation" is done in 6.4.

6.4 Analysis

Figure 5 presents the colorization results of our different algorithms on the same validation video. The images in the figure were sampled from the video every 7 frames.

To generate the fourth video (fourth row), our first conditional GAN model was applied to each grayscale frame conditioned on the previous ground truth color frame. This re-colorized video is almost indistinguishable from the ground truth. This shows that the models achieve good color transfer between frames.

However, as seen on the next rows, the performances of the GAN models G quickly decline when using them for color propagation over the whole video. The videos were colorized step-by-step as follows: at time t, the grayscale frame g_t was colorized with $\hat{c}_t = G(g_t, \hat{c}_{t-1})$ (or $\hat{c}_t = G(g_t, \hat{c}_{t-1}, c_1)$ for the extended conditional GAN model), where $\hat{c}_{t-1}$ is the previous re-colorized frame.

The models used to generate the fifth, sixth and seventh rows are our first conditional GAN model trained with ($dt = 1$) consecutive frames (e), our first conditional GAN model trained with ($dt = 5$) consecutive frames (f), and our extended conditional GAN model trained with ($dt = 5$) consecutive frames (f) respectively.

The first issue is that consecutive frames are very highly correlated. When training on consecutive frames ($dt = 1$), the conditional GAN model can essentially learn to copy-paste colors to achieve a very good pixel accuracy. However, when we apply color propagation throughout a video, this causes colorization errors on fast moving objects. This can be for example seen on the second image of the fifth row. Even tough the left hand of the front girl moved, the algorithm colorizes it in gray: in the color of the background which occupied this pixels in the first frame.

To try to solve this issue, our network was retrained on more time-spaced frames: ($dt = 5$) consecutive frames. This helped as can be seen on the second image of the sixth row. This time, the algorithm managed to track the hand of the girl which is now correctly colorized in pink.

However, this approach still could not solve the problem of **exponential error propagation**. This error is caused by the distribution difference between training examples and what the GAN has to work with for color propagation. Indeed, at test time, the conditional GAN must colorize the grayscale frame g_t conditioned on the ground truth color frame c_{t-dt}. However when doing color propagation, the GAN must colorize the grayscale frame g_t conditioned on the previous re-colorized frame $\hat{c}_{t-1}$. This means that if there are some mistakes in the colorization of frame $\hat{c}_{t-1}$, this wrong colorization will be considered as the ground truth to colorize the next frame. This can be seen for example in

Figure 5: Comparison of Video Colorizations. From top row to bottom: (a) grayscale ground truth, (b) color ground truth, (c) baseline frame-by-frame colorization, (d) GAN colorization from ground truth reference, (e) GAN color propagation (trained with $dt = 1$), (f) GAN color propagation (trained with $dt = 5$) (g) extended GAN color propagation (trained with $dt = 5$)

the sixth row where the blue color mistake gets amplified and contaminates the colorization of all neighbouring pixels.

Quantitatively if the pixel accuracy for color transfer is $(1 - \eta)$, then after t steps of color propagation the pixel accuracy of the t-th frame is on the order of $(1-\eta)^t$: hence the exponential error propagation.

This is what we tried to solve with our last "extended" conditional GAN model, where an additional conditioning on the first color image was introduced. The idea was that even if the colorization makes some mistakes, at the next step the algorithm would still have the first color image as a reference "color palette" to colorize the next image in a realistic fashion.

However, the last row shows that it did not succeed. The network did not learn to identify regions well, and we still have the problem of copy-pasted colors. This can be seen for example on the sweater of the front girl: the sweater moved, but the colors did not.

7 Conclusion

7.1 Summary

We successfully trained a GAN-based model for color transfer in grayscale videos. Our model achieves high pixel accuracy, and plausibility for color transfer between consecutive frames. However this performance declines quickly when we use this model to perform end-to-end color propagation due to **exponential error propagation** throughout the video.

We tried extending our model by including an extra conditioning on a color reference image, but that did not improve the performances.

Finally, we considered some avenues for further work, such as extending the generator network with a similarity based sub-net or deploying a convolutional RNN architecture in order to process sequences of images directly.

7.2 Future Work

Unfortunately adding an additional base reference image did not improve the performance of the algorithm. In fact, with the extended conditional GAN model, we got worse quantitative and qualitative results. After examining some generated samples it seems that the fundamental issue was that our generator network was not able to map colors from the reference image to the output correctly.

In their recent paper, [HCL$^+$18] managed to colorize grayscale images using a reference color image that did not have very high degree of content correlation with the target image. The network architecture used in that paper was separated in two parts: a similarity sub-net that computed distances between regions of the two images and a colorization sub-net that applied color based on these distances.

Our generator architecture does not have a dedicated distance encoding module. An approach that we thought would be worth exploring was to extend the model with a such a module. This similarity sub-net could help to leverage the color distribution of the base reference to consistently colorize the whole video.

8 Codebase

All our work is available on the following GitHub repository: `https://github.com/ColasGael/Automatic-Video-Colorization`

9 Oral Presentation

An oral presentation of this final project can be found on Youtube: `https://www.youtube.com/watch?v=SSyEcJI6b3U&feature=youtu.be`

This link can be shared on the CS236 website.

References

[AR15] Soumith Chintala Alec Radford, Luke Metz. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015. https://arxiv.org/abs/1511.06434.

[HCL$^+$18] Mingming He, Dongdong Chen, Jing Liao, Pedro V. Sander, and Lu Yuan. Deep exemplar-based colorization. *ACM Trans. Graph.*, 37(4):47:1–47:16, July 2018.

[Lab] MIT-IBM Watson AI Lab. "moments in time" dataset. http://moments.csail.mit.edu/.

[LLW04] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, August 2004.

[NNE18] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization using generative adversarial networks. In *International Conference on Articulated Motion and Deformable Objects*, pages 85–94. Springer, 2018.

[pac17] Open CV Python package. Open cv - getting started with videos, 2017. https://docs.opencv.org/3.4.0/dd/d43/tutorial_py_video_display.html.

[VSF+18] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. 2018. https://arxiv.org/pdf/1806.09594.

[Yeu17] Xingjian Shi Zhourong Chen Hao Wang Dit-Yan Yeung. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *NIPS 2017*, 2017.

[ZIE16] Richard Zhang, Phillip Isola, and Alexei Efros. Colorful image colorization. 9907:649–666, 10 2016. https://github.com/richzhang/colorization.

[ZYL12] Zhong Zhen, Gui Jun Yan, and Ma Lizhuang. An automatic image and video colorization algorithm based on pattern continuity. *2012 International Conference on Audio, Language and Image Processing*, pages 531–536, 2012.